

# Richiami di linguaggio C++

- **Struttura di un file sorgente:**

```
#include <file intestazione>
#direttive al preprocessore
dichiarazioni globali;
int main() {
    dichiarazioni locali;
    istruzioni;
    funzioni(parametri);
    return intero;
}
```

- **file d'intestazione**  
Contengono le definizioni delle funzioni. I più comuni sono *iostream* per fare input e output, *iomanip* per formattare l'output, *cmath* per usare le funzioni matematiche.
- **dichiarazione** di funzioni e variabili a livello di file  
Prima di *main* definisco le variabili e le funzioni visibili da tutto il programma
- **programma principale:** `int main(argc, argv[])`
- **tipi principali di dati:** `int` e `long`, `float` e `double`, `char`
- **puntatori:** sono gli indirizzi delle variabili e degli array: se `p[100]` è un array di interi, `p` è un puntatore a intero
- **funzioni**
  - ritornano un valore (se non `void`)
  - gli argomenti sono passati per valore, per puntatore o per riferimentoSe la funzione deve modificare il valore dei propri argomenti, questi devono essere passati come puntatori o riferimenti alle variabili da cambiare.
- **operatori fondamentali** `=, +, -, /, *`

# Strutture fondamentali

Due strutture costituiscono il 90 per cento dei programmi scientifici

## cicli

```
for(int j=0; j<10; j++) {  
    cout<<"j vale "<<j<<endl;  
}
```

Altri cicli sono `while(condizione){...}` e `do{...}while(condizione)`

Per uscire da un ciclo si usa `break`

Per passare alla prossima iterazione del ciclo si usa `continue`

## istruzioni condizionali

```
if(j==3) {  
    cout<<"J vale 3\n";  
}  
else if(j==2) {  
    cout<<"J vale 2\n";  
}  
else cout<<"J non vale 2 o 3\n";
```

Un'altra istruzione condizionale è `switch...case`:

Le condizioni sono espresse tramite gli operatori

`==, !=, <, >, <=, >=`

**nota bene:** `==` non è `=`

## Alcune funzioni matematiche

$x^y$	=	<code>pow(x,y)</code>
$\sqrt{x}$	=	<code>sqrt(x)</code>
$e^x$	=	<code>exp(x)</code>
$\sin(x)$	=	<code>sin(x)</code>
$\cos(x)$	=	<code>cos(x)</code>
$\tan(x)$	=	<code>tan(x)</code>
$\sin^{-1}(x)$	=	<code>asin(x)</code>
$\cos^{-1}(x)$	=	<code>acos(x)</code>
$\tan^{-1}(x)$	=	<code>atan(x)</code>
<code>floor(x)</code>	=	massimo intero che non supera x
<code>fabs(x)</code>	=	$ x $

- Un'altra operazione importante è il resto della divisione per un numero
- Per gli interi `i%5` da' il resto della divisione di i per 5
- Per i reali `fmod(x,3.14)` da' il resto della divisione di x per 3.14.
- Per usare tutte queste funzioni bisogna includere `<cmath>` e linkare `-lm`

# Costanti

Un modo per ottenerle è

$$\begin{aligned}\pi &= 4 \cdot \arctan(1.) \\ e &= \exp(1.)\end{aligned}$$

Uno più standard è guardare le definizioni nel file math.h

$$\begin{aligned}\pi &= M\_PI \\ e &= M\_E \\ \sqrt{2} &= M\_SQRT2\end{aligned}$$

## Overloading delle funzioni

Una funzione è definita dal valore che ritorna, dal numeri e dal tipo degli argomenti. Le due funzioni seguenti non coincidono

```
int square(int x)
{
    return x*x;
}
```

```
double square(double x)
{
    return x*x;
}
```

Anche se il codice è identico, le funzioni hanno un diverso tipo di parametri e di valore ritornato, e vengono considerate diverse dal compilatore: questa caratteristica si chiama *overloading* (*sovraccarico*)

Un esempio di funzioni sovraccariche sono gli operatori, come la somma, che possono operare su tipi diversi. Definendo nuovi tipi di dati è possibile definire in modo personalizzato la somma, la differenza, il prodotto di due dati.

## Classi

Le classi sono forse il maggior punto di forza del `C++` perché permettono una più chiara organizzazione del codice. Una classe è un contenitore in cui possono stare due tipi di oggetti: i dati e i metodi. Per fare un esempio, la classe "Torta di mele" contiene la farina, l'acqua, le mele e lo zucchero, più il metodo per fare l'impasto e il metodo per cuocere. Se volete fare una buona torta di mele `C++`, non potete accedere direttamente alla farina e agli altri ingredienti (che si dicono *private* ma potete impastarla e cuocerla (diciamo che questi metodi sono *public*). Se  $x$  e  $y$  sono entrambi torte di mele, diciamo che ciascuno è un'istanza della classe.